Juniper For Fun and Profit

[TOC]

Introduction

Juniper is a framework within Unity 3D for managing and developing virtual, augmented, and mixed reality applications, within the context of a metaphor of the human body. It provides the following sub-modules:

- *Animation* - Basic animations on specific, common object properties, without having to engage the full power of Mecanim.
- *Audio* - Audio management and effects components.
- *Data* - Managing HTTP requests and streaming assets, all from background threads that do not lock up the main UI thread.
- *Editor* - Custom editors for Juniper's Unity Components to make them easier to edit within the Unity Editor. Also includes the Unit Tests for all other code modules.
- *Extensions* - Useful functions to add on top of Unity and .NET's existing classes.
- *Haptics* - Components for performing high-quality haptic feedback on systems that support it.
- *Imaging* - Loading and processing PNG and raw images.
- *Input* - Handling user input: keyboard, mouse, touch, motion controllers, and speech.
- *Light Mapping* - Swapping light map sets at runtime to have different, baked lighting conditions without switching scenes.
- *Statistics* - Collections that perform statistical analysis on their contents.
- *Widgets* - Visual elements with which the user may interact.
- *World* - Components for understanding and integrating with the real world: light estimation, global positioning, translating between spatial reference systems, receiving weather reports.
- *Units* - Conversions between and formattings of values tagged with units of measure.
- *XR* - Management of VR and AR hardware, specifically the detection of surfaces and the tracking of positional hardware.
- *Other* - Scene and project management tools, for managing complex scene hierarchies in the Unity editor and at runtime.

##Juniper

```
───────────────────────────────────────────────────────────────────────────────
Language                 Files     Lines     Code  Comments   Blanks Complexity
───────────────────────────────────────────────────────────────────────────────
C#                         561     64677    40146     15931     8600       4774
XML                         21    148903   133227         0    15676          0
Plain Text                   2        22       18         0        4          0
JSON                         2       311      311         0        0          0
License                      2        32       26         0        6          0
Objective C++                1       107       64        35        8         20
───────────────────────────────────────────────────────────────────────────────
Total                      589    214052   173792     15966    24294       4794
───────────────────────────────────────────────────────────────────────────────
Estimated Cost to Develop $11,874,999
Estimated Schedule Effort 30.455721 months
Estimated People Required 23.632647
───────────────────────────────────────────────────────────────────────────────
```

Setup

Installation

There are two ways to install Juniper.

- [Download the UnityPackage (Juniper.unitypackage)] and install it in your project, or
- [Clone the repository (https://github.com/capnmidnight/Juniper)] and copy the contents of the `Package` directory to your Assets folder in a directory named `Juniper`.

Project Integration

There are a number of different components that go into configuring a Juniper application:

- The AR or VR subsystem needs to be setup for use.
- The EventSystem needs to be configured,
    - along with an InputModuel that understands different types of 3D pointers, and,
    - those pointers appropriate for the XR subsystem.

- The user-interface interaction feedback audio system, which includes haptic feedback.
- A manager for AR targets, if they are being used.
- Mouse and Keyboard movement for use in the Editor.
- A camera configuration that,
    - updates the clear flags when switching between types of XR subsystems, and
    - reduces rendering quality to keep the target frame-rate as close to the native display refresh rate as possible.
- And, a graphical component for fading views in and out during scene transitions and other long, blocking operations, so the user doesn't see the camera locked in place.

To that end, there is a Prefab that has most of this pre-configured, requiring only project-specific configuration once included in the project. It is located at

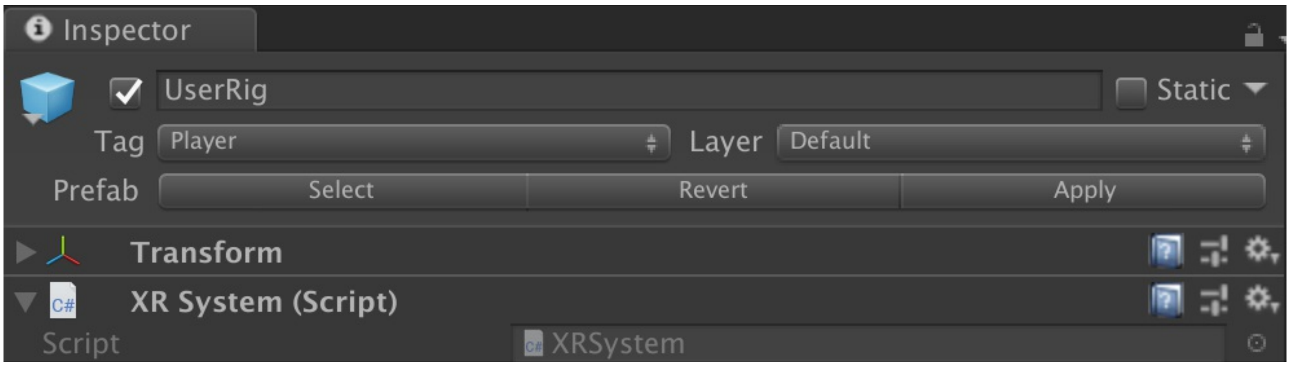
```
Juniper/Prefabs/UserRig.prefab
```

The prefab contains the following GameObjects and Components:

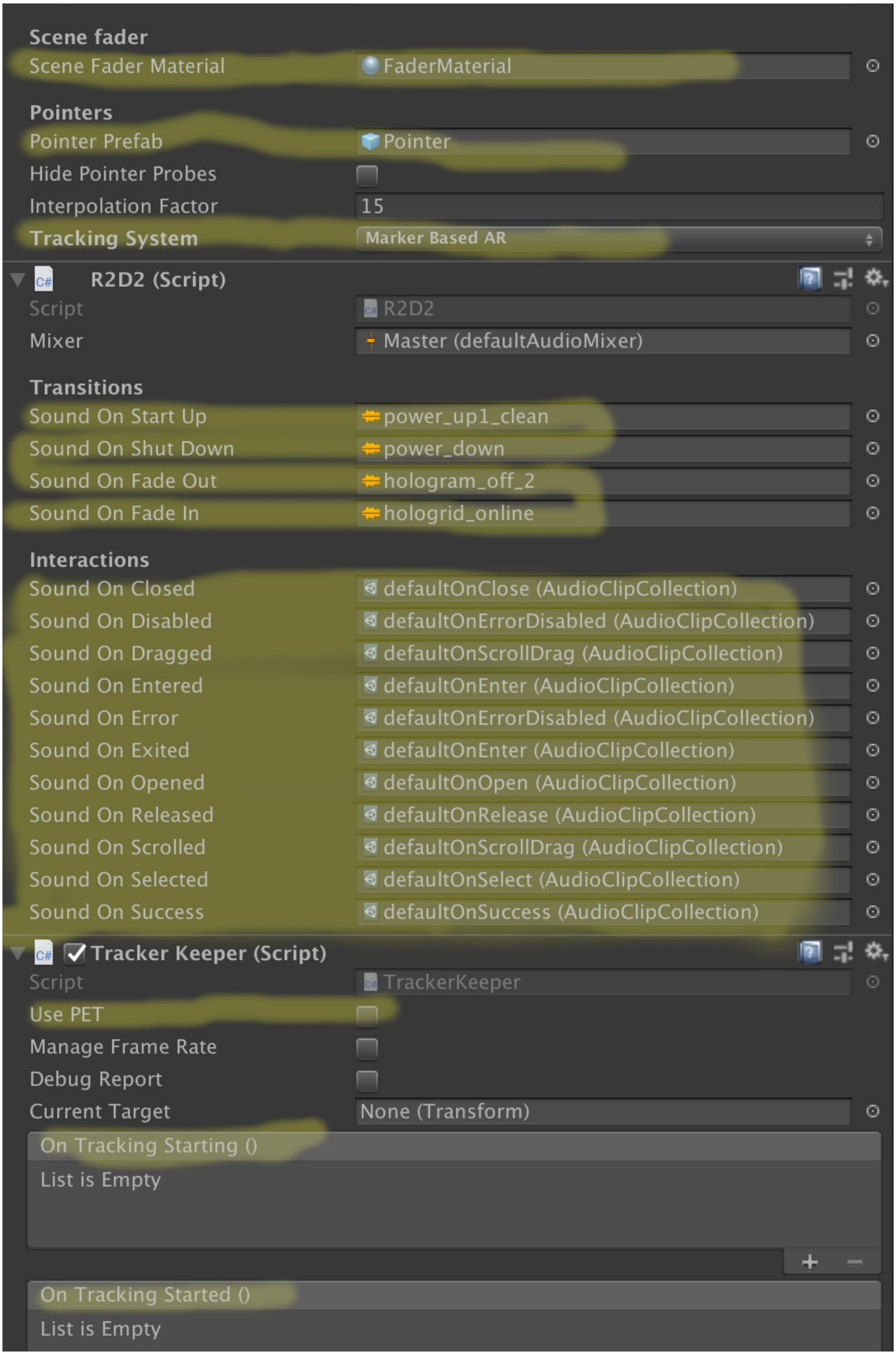
- **[GO] UserRig**: The root of the prefab
    - *[CMP] XRSystem*: manages system configuration and AR/VR feature sets.
    - *[CMP] EventSystem*: Unity's EventSystem component for triggering pointer events.
    - *[CMP] UnifiedInputModule*: A custom InputModule that fuses a variety of different pointers between the computer mouse, the touch screen, and any potential motion controllers or hand tracking systems connected to the application.
    - *[CMP] Mouse*: The mouse pointer, mostly only used in the Editor.
    - *[CMP] Touches*: Invisible pointers for each finger.
    - *[CMP] TrackerKeeper*: A top-level manager for AR image targets, with event handlers for a variety of different scenarios in managing the AR experience.
    - *[CMP] R2D2*: Configuration setting for different audio clips to playback during different types of user interactions in the system. This also provides haptic feedback for most of the interaction types.
    - **[GO] Stage**: The immediate area around the user, separate from the camera. Keyboard/GamePad movement, or VR teleportation will move the stage instead of the camera. Moving the camera is reserved for the XR subsystem, to maintain the right view angle for the user.
        - *[CMP] KeyboardMovement*: Moves the stage with Unity's configured keyboard movement keys in the direction the camera is looking. Mostly only used in the Editor for testing.
        - **[GO] Camera**: The main camera.
            - *[CMP] Camera*: The main camera component. You may want to configure the Clipping Planes. Do not change the Field of View, as it will be ignored and recomputed at runtime for the specific XR subsystem.
            - *[CMP] CameraExtensions*: Manages camera attributes like camera FOV.
            - *[CMP] QualityDegrader*: Projects start out at the highest quality setting and degrade through the quality settings enabled in the Editor until either there are no more quality setting to go through or the app consistently hits 60FPS.
            - *[CMP] PhysicsRaycaster*: The raycaster used for all pointers that fire through the main camera, including the Moues and Touches.
            - **[GO] Fader**: A box that sits around the user's head for fading the camera view in and out in a way that works on all XR subsystems.
                - *[CMP] Darth*: Darth Fader is the component that manages fading of the camera view. You can retrieve Darth Fader and use it to fade out the screen if you ever have any long, blocking operations that could lock up the camera.

Configuration

Once you have the UserRig prefab in your scene, select the UserRig game object and find the XRSystem component. Pay the most attention to the highlighted fields:



UserRig
Tag Player
Layer Default
Prefab
Script XRSystem



Scene fader

Scene Fader Material	FaderMaterial

Pointers

Pointer Prefab	Pointer
Hide Pointer Probes	☐
Interpolation Factor	15
Tracking System	Marker Based AR

R2D2 (Script)

Script	R2D2
Mixer	Master (defaultAudioMixer)

Transitions

Sound On Start Up	power_up1_clean
Sound On Shut Down	power_down
Sound On Fade Out	hologram_off_2
Sound On Fade In	hologrid_online

Interactions

Sound On Closed	defaultOnClose (AudioClipCollection)
Sound On Disabled	defaultOnErrorDisabled (AudioClipCollection)
Sound On Dragged	defaultOnScrollDrag (AudioClipCollection)
Sound On Entered	defaultOnEnter (AudioClipCollection)
Sound On Error	defaultOnErrorDisabled (AudioClipCollection)
Sound On Exited	defaultOnEnter (AudioClipCollection)
Sound On Opened	defaultOnOpen (AudioClipCollection)
Sound On Released	defaultOnRelease (AudioClipCollection)
Sound On Scrolled	defaultOnScrollDrag (AudioClipCollection)
Sound On Selected	defaultOnSelect (AudioClipCollection)
Sound On Success	defaultOnSuccess (AudioClipCollection)

✓ Tracker Keeper (Script)

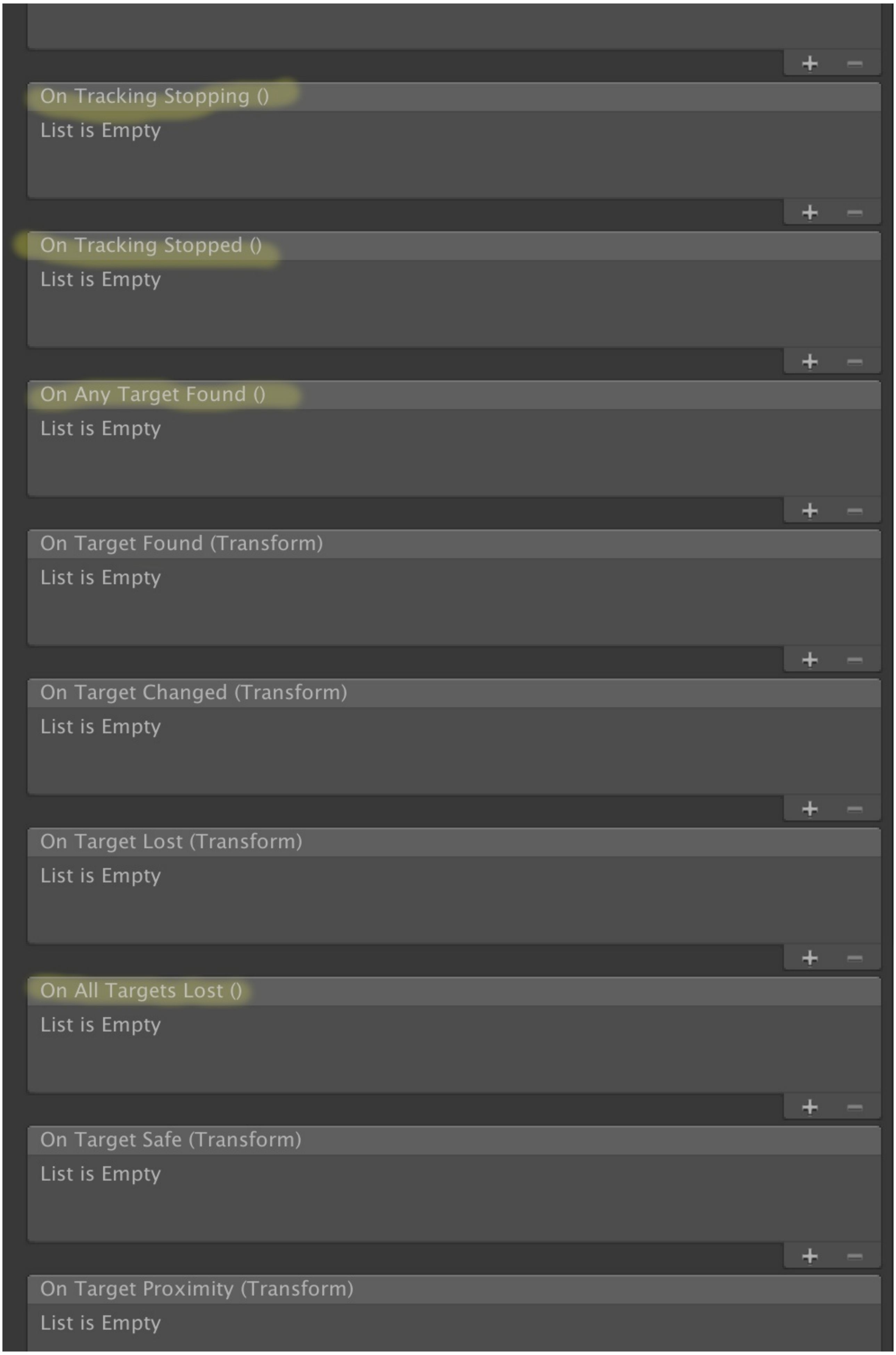
Script	TrackerKeeper
Use PET	☐
Manage Frame Rate	☐
Debug Report	☐
Current Target	None (Transform)

On Tracking Starting ()

List is Empty

On Tracking Started ()

List is Empty


On Tracking Stopping ()
List is Empty

On Tracking Stopped ()
List is Empty

On Any Target Found ()
List is Empty

On Target Found (Transform)
List is Empty

On Target Changed (Transform)
List is Empty

On Target Lost (Transform)
List is Empty

On All Targets Lost ()
List is Empty

On Target Safe (Transform)
List is Empty

On Target Proximity (Transform)
List is Empty

Notes:

- Make sure to set the TrackingSystem on XRSystem. It will initially be set to None.
- Additionally, if this is the first time you are setting up a Vuforia-based project and you select the "Marker Based AR" tracking type, you may need to use the GameObject menu in Unity to add a Vuforia prefab in order to force the Vuforia components to install into your project. You may delete it afterwards.
- R2D2 will be configured with a default set of sounds that come with Juniper. These sounds are Star Trek sound effects, so they will not be appropriate to deploy to a production system. Make sure you find a new set of sounds appropriate for your application.
- TrackerKeeper has a number of interesting events for managing your project workflow. It's recommended that you run Vuforia in Delayed Initialization mode, so your application can start with a splash-screen/menu view without the camera constantly running an draining the battery.

Development

Clone the repository and copy the `Package/` directory to one of the `Test/<PlatformName>Test/Assets` directories.

Tools

- Unity Hub v0.20.1 (https://blogs.unity3d.com/2018/01/24/streamline-your-workflow-introducing-unity-hub-beta/) (optional). Unity Hub is very handy for managing different versions of Unity between different projects.

- Unity Editor v2018.2.0f2 (https://unity3d.com/get-unity/download/archive). Unity 2018 has a package manager available that makes it easier to keep projects and their dependencies in sync without having to save all of the dependencies in the repository.

- Visual Studio for Mac, Community Edition v7.5.2 (https://visualstudio.microsoft.com/vs/mac/). Any editor is probably fine.

- **Vuforia v7.1.35**. This is the default version that comes with Unity 2018.2.01f2.

Debugging

The prefab at:

```
Juniper/Prefabs/DebugUI.prefab
```

Contains a Canvas and Text element that is used by the various components to print out their status.

GPSLocation, SunPosition, Weather, and OutdoorLightEstimate all have boolean values controlling whether or not they provide real world values versus static values defined at design time. That boolean is called `Fake <X>`, where X is whatever property is being faked. They also include a boolean called `Print Debug Report` that enables showing the current status of the component on the ScreenDebugger.

Platform Notes

These are notes specific to each AR subsystem and/or operating system

Hololens

Build settings

The Hololens documentation talks about a Unity Build setting "C# Projects" that enables completing the build and deploying to the device from Visual Studio rather than from Unity. The rationale for this feature is that it makes it faster to iterate on *changes* to scripts. If you want to make changes to the scene, to any assets, or add any scripts, you need to still do it through Unity.

As of Version 2017.2 of Unity, this feature is broken, defaulting the Build Target Architecture to ARM. Just manually change the architecture to x86 (not x64).